MACHINE LEARNING

ASSIGNMENT - 5

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

Answer: R-squared is generally a better measure of the goodness of fit for a regression model than the residual sum of squares (RSS).

The reason why R-squared is often preferred over RSS as a measure of goodness of fit is due to its standardized nature:

1. Scalability: R-squared is scale-invariant, meaning it does not change if the scale of the data changes, whereas RSS is affected by the scale of the dependent variable. This makes R-squared a better choice when comparing models fitted on different scales.

2. Interpretability: R-squared has an intuitive interpretation as the proportion of variance explained, which is easier to understand than the sum of squared residuals. An R-squared of 0.75 means that 75% of the variance in the dependent variable is explained by the model, which is a straightforward interpretation.

3. Benchmarking: R-squared provides a clear benchmark. An R-squared of 0 indicates that the model explains none of the variability in the response data around its mean, while an R-squared of 1 indicates that the model explains all the variability.

4. Adjustment for model complexity: Adjusted R-squared takes into account the number of predictors in the model, which helps in assessing whether the addition of a new predictor really improves the model or is just adding complexity without significantly improving the fit.

2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

Answer:

TSS = $\sum(Yi - \bar{Y})^2$, where Yi is the actual value of the response variable for observation i, and $\bar{Y}$ is the mean of the response variable

ESS = $\sum(\hat{Y}i - \bar{Y})^2$, where $\hat{Y}i$ is the predicted value of the response variable for observation I and $\bar{Y}$ is the mean of the response variable.

RSS = $\sum(Yi - \hat{Y}i)^2$, which is the sum of squared differences between the actual and predicted values of the response variable.

Equation for relating 3 metrics is:

TSS=ESS+RSS

3**. What is the need of regularization in machine learning?**

Answer: Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

4. What is Gini–impurity index?

Answer: The Gini Index is a proportion of impurity or inequality in statistical and monetary settings. In machine learning, it is utilized as an impurity measure in decision tree algorithms for classification tasks. With regards to decision trees, the Gini Index is utilized to determine the best feature to split the data on at every node of the tree.The Gini Index measures the probability of a haphazardly picked test being misclassified by a decision tree algorithm, and its value goes from 0 (perfectly pure) to 1 (perfectly impure).

5. Are unregularized decision-trees prone to overfitting? If yes, why?

Answer: Yes, the unregularized decision-trees are prone to overfitting due to following reasons:

1. **Flexibility of the model:** Decision trees can create very complex, non-linear decision boundaries that closely fit the training data. As the tree grows deeper, it can create more and more intricate splits to perfectly classify the training examples.
2. **Greedy nature of tree building:** Decision tree algorithms typically use a greedy approach, making locally optimal decisions at each node to maximize some metric like information gain or Gini impurity. This can lead to the tree becoming overly complex and specialized to the training data.
3. **Lack of regularization:** Basic decision tree algorithms do not have strong built-in regularization mechanisms. Things like tree depth, minimum samples per leaf, or maximum features per split are often not tuned rigorously, allowing the tree to grow too complex.
4. **Noise sensitivity:** Decision trees can be very sensitive to small variations or noise in the training data. Spurious patterns in the training set can lead the algorithm to create splits that overfit and do not generalize well.

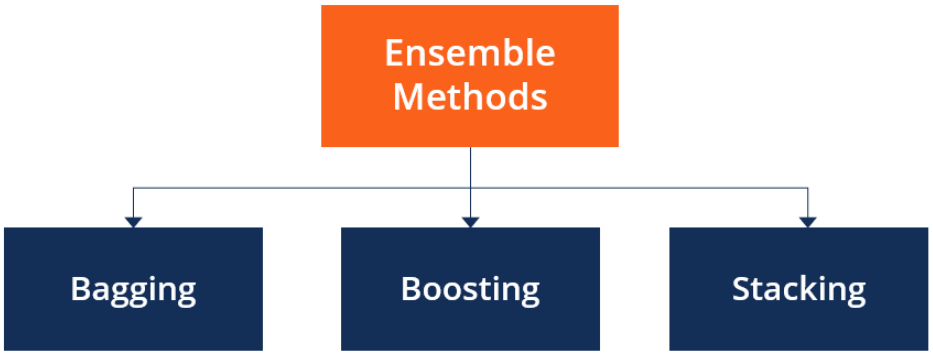
The end result is a decision tree model that performs well on the training data but fails to generalize to new, unseen examples - a classic case of overfitting. Techniques like pruning, setting depth limits, and using ensemble methods like random forests can help mitigate this tendency to overfit in decision trees.

6.What is an ensemble technique in machine learning?

Answer: Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model. The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine learning.

Ensemble methods aim at improving predictability in models by combining several models to make one very reliable model.

Ensemble methods are ideal for regression and classification, where they reduce bias and variance to boost the accuracy of models.

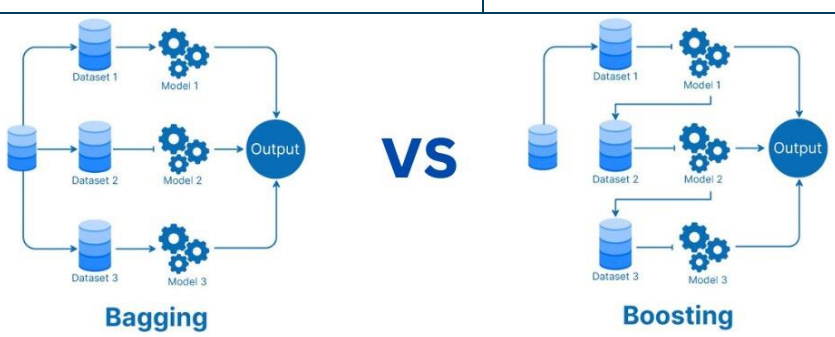


7. What is the difference between Bagging and Boosting techniques?

Answer: Bagging and boosting are different ensemble techniques that use multiple models to reduce error and optimize the model. The bagging technique combines multiple models trained on different subsets of data, whereas boosting trains the model sequentially, focusing on the error made by the previous model.

Difference Between Bagging and Boosting: Bagging vs Boosting

	Bagging	Boosting
Basic Concept	Combines multiple models trained on different subsets of data.	Train models sequentially, focusing on the error made by the previous model.
Objective	To reduce variance by averaging out individual model error.	Reduces both bias and variance by correcting misclassifications of the previous model.
Data Sampling	Use Bootstrap to create subsets of the data.	Re-weights the data based on the error from the previous model, making the next models focus on misclassified instances.
Model Weight	Each model serves equal weight in the final decision.	Models are weighted based on accuracy, i.e., better-accuracy models will have a higher weight.
Error Handling	Each model has an equal error rate.	It gives more weight to instances with higher error, making subsequent model focus on them.
Overfitting	Less prone to overfitting due to average mechanism.	Generally not prone to overfitting, but it can be if the number of the model or the iteration is high.
Performance	Improves accuracy by reducing variance.	Achieves higher accuracy by reducing both bias and variance.
Common Algorithms	Random Forest	AdaBoost, XGBoost, Gradient Boosting Mechanism
Use Cases	Best for high variance, and low bias models.	Effective when the model needs to be adaptive to errors, suitable for both bias and variance errors.



8. What is out-of-bag error in random forests?

Answer: Out of bag (OOB) score is a way of validating the Random forest model. Below is a simple intuition of how is it calculated followed by a description of how it is different from validation score and where it is advantageous.For the description of OOB score calculation, let's assume there are five DTs in the random forest ensemble labeled from 1 to 5. For simplicity, suppose we have a simple original training data set as below.

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Let the first bootstrap sample is made of the first three rows of this data set as shown in the green box below. This bootstrap sample will be used as the training data for the DT "1".

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Bootstrap sample

Then the last row that is "left out" in the original data (see the red box in the image below) is known as Out of Bag sample. This row will not be used as the training data for DT 1. Please note that in reality there will be several such rows which are left out as Out of Bag, here for simplicity only one is shown.

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Out of Bag sample

After the DTs models have been trained, this leftover row or the OOB sample will be given as unseen data to the DT 1. The DT 1 will predict the outcome of this row. Let DT 1 predicts this row correctly as "YES". Similarly, this row will be passed through all the DTs that did not contain this row in their bootstrap training data. Let's assume that apart from DT 1, DT 3

and DT 5 also did not have this row in their bootstrap training data. The predictions of this row by DT 1, 3, 5 are summarized in the table below.

Decision Tree	Prediction
1	YES
3	NO
5	YES
Majority vote : YES	

We see that by a majority vote of 2 “YES” vs 1 “NO” the prediction of this row is “YES”. It is noted that the final prediction of this row by majority vote is a **correct prediction** since originally in the “Play Tennis” column of this row is also a “YES”.

Similarly, each of the OOB sample rows is passed through every DT that did not contain the OOB sample row in its bootstrap training data and a majority prediction is noted for each row.

And lastly, the OOB score is computed as **the number of correctly predicted rows from the out of bag sample.**

9. What is K-fold cross-validation?
Answer: K-fold cross validation
In K-fold cross-validation, the data set is divided into a number of K-folds and used to assess the model’s ability as new data become available. K represents the number of groups into which the data sample is divided. For example, if you find the k value to be 5, you can call it 5-fold cross-validation. Each fold is used as a test set at some point in the process.

1.Randomly shuffle the dataset.

2.Divide the dataset into k folds

3.For each unique group:

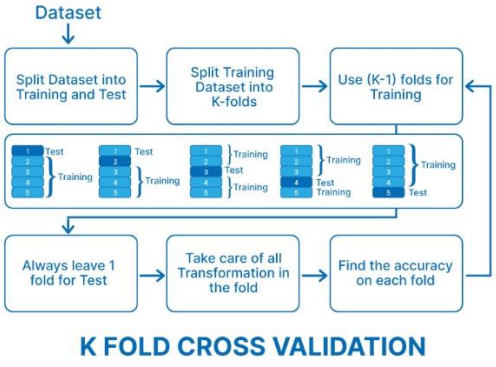
-Use one fold as test data

-Use remaining groups as training dataset

-Fit model on training set and evaluate on test set

-Keep Score

4. Get accuracy score by applying mean to all the accuracies received for all folds.



As you can see in the fig that there is a dataset which is Divided into 5 folds. That means there will be five iterations, and in each iteration, there will be one test fold, and the other four folds will be training folds. And in each iteration, test and training folds keep on changing. That means if we have 1000 records in our data set, then suppose 200 records are our test data, and 800 records are our training data.

So in the first iteration, (1-200) records will be **test data,** and (201-1000) will be **training data**. In the second iteration, (1-200) records plus (401-1000) represent **training data**, And (200 -400) will represent the **test data.** And so on…

10. What is hyper parameter tuning in machine learning and why it is done?

Answer: When you're training machine learning models, each dataset and model needs a different set of hyperparameters, which are a kind of variable. The only way to determine these is through multiple experiments, where you pick a set of hyperparameters and run them through your model. This is called hyperparameter tuning. In essence, you're training your model sequentially with different sets of hyperparameters. This process can be manual, or you can pick one of several automated hyperparameter tuning methods.

Whichever method you use, you need to track the results of your experiments. You'll have to apply some form of statistical analysis, such as the loss function, to determine which set of hyperparameters gives the best result. Hyperparameter tuning is an important and computationally intensive process.

Hyperparameter tuning is important because:
Hyperparameters directly control model structure, function, and performance. Hyperparameter tuning allows data scientists to tweak model performance for optimal results. This process is an essential part of machine learning, and choosing appropriate hyperparameter values is crucial for success.

For example, assume you're using the learning rate of the model as a hyperparameter. If the value is too high, the model may converge too quickly with suboptimal results. Whereas if the rate is too low, training takes too long and results may not converge. A good and balanced choice of hyperparameters results in accurate models and excellent model performance.

11. What issues can occur if we have a large learning rate in Gradient Descent?

Answer: The learning rate is an important hyperparameter that greatly affects the performance of gradient descent. It determines how quickly or slowly our model learns, and it plays an important role in controlling both convergence and divergence of the algorithm. When the learning rate is too large, gradient descent can suffer from divergence. This means that weights increase exponentially, resulting in exploding gradients which can cause problems such as instabilities and overly high loss values. On the other hand, if the learning rate is too small, then gradient descent can suffer from slow convergence or even stagnation—which means it may not reach a local minimum at all unless many iterations are performed on large datasets.

12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?
Answer: We cannot use Logistic Regression for classification of Non-Linear Data because Logistic Regression assumes a linear relationship between the input features and the output. This means that it cannot capture the complexity and non-linearity of the data. Another drawback is that it is sensitive to outliers and noise, which can affect the accuracy and stability of the model. Logistic regression also has a limited capacity to learn from multiple features, as it can only combine them linearly.

13. Differentiate between Adaboost and Gradient Boosting.

Answer: The most significant difference is that gradient boosting minimizes a loss function like MSE or log loss while AdaBoost focuses on instances with high error by adjusting their sample weights adaptively.

Gradient boosting models apply shrinkage to avoid overfitting which AdaBoost does not do. Gradient boosting also performs subsampling of the training instances while AdaBoost uses all instances to train every weak learner.

Overall gradient boosting is more robust to outliers and noise since it equally considers all training instances when optimizing the loss function. AdaBoost is faster but more impacted by dirty data since it fixates on hard examples

14. **What is bias-variance trade off in machine learning?**
Answer: The bias-variance tradeoff is about finding the right balance between simplicity and complexity in a machine learning model. High bias means the model is too simple and consistently misses the target, while high variance means the model is too complex and shoots all over the place.

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Answer: In Support Vector Machines (SVM), kernels are functions used to map the input data into higher-dimensional space to make it easier to find a separating hyperplane. Here's a brief overview of three common types of kernels: linear, Radial Basis Function (RBF), and polynomial.

1. Linear Kernel

The linear kernel is the simplest form of kernel.

Characteristics:

- It does not transform the data; it simply uses the original feature space.
- Best used when the data is already linearly separable or nearly linearly separable.
- Computationally less intensive compared to other kernels.

Example Use Case: When the relationship between data points is approximately linear, or the dataset has a very high number of features.

2. Radial Basis Function (RBF) Kernel

The RBF kernel, also known as the Gaussian kernel

Characteristics:

- Maps data into an infinite-dimensional space.
- Effective in capturing complex relationships in the data.
- The parameter γ\gammaγ controls the range of influence of a single training example. A small γ\gammaγ means a large influence, while a large γ\gammaγ means a small influence.

Example Use Case: When the data is not linearly separable and has a more complex decision boundary.

3. Polynomial Kernel

Characteristics:

- Transforms the data into a higher-dimensional space where polynomial decision boundaries can be constructed.
- The degree ddd controls the complexity of the decision boundary.
- For d=1d = 1d=1, it is equivalent to the linear kernel. For d=2d = 2d=2, it introduces quadratic interactions between features.

Example Use Case: When interactions between features need to be captured, and you expect polynomial relationships between the features and the output.

Choosing a Kernel

- **Linear Kernel:** Use when the data is linearly separable or nearly so.
- **RBF Kernel:** Use for most non-linear problems, especially when you have no prior knowledge of the data distribution.
- **Polynomial Kernel:** Use when you believe that the decision boundary can be captured by polynomial interactions between features.

Each kernel has its strengths and is suited for different types of data and problems. Experimenting with different kernels and tuning their parameters is often necessary to find the best performing model for a specific problem.